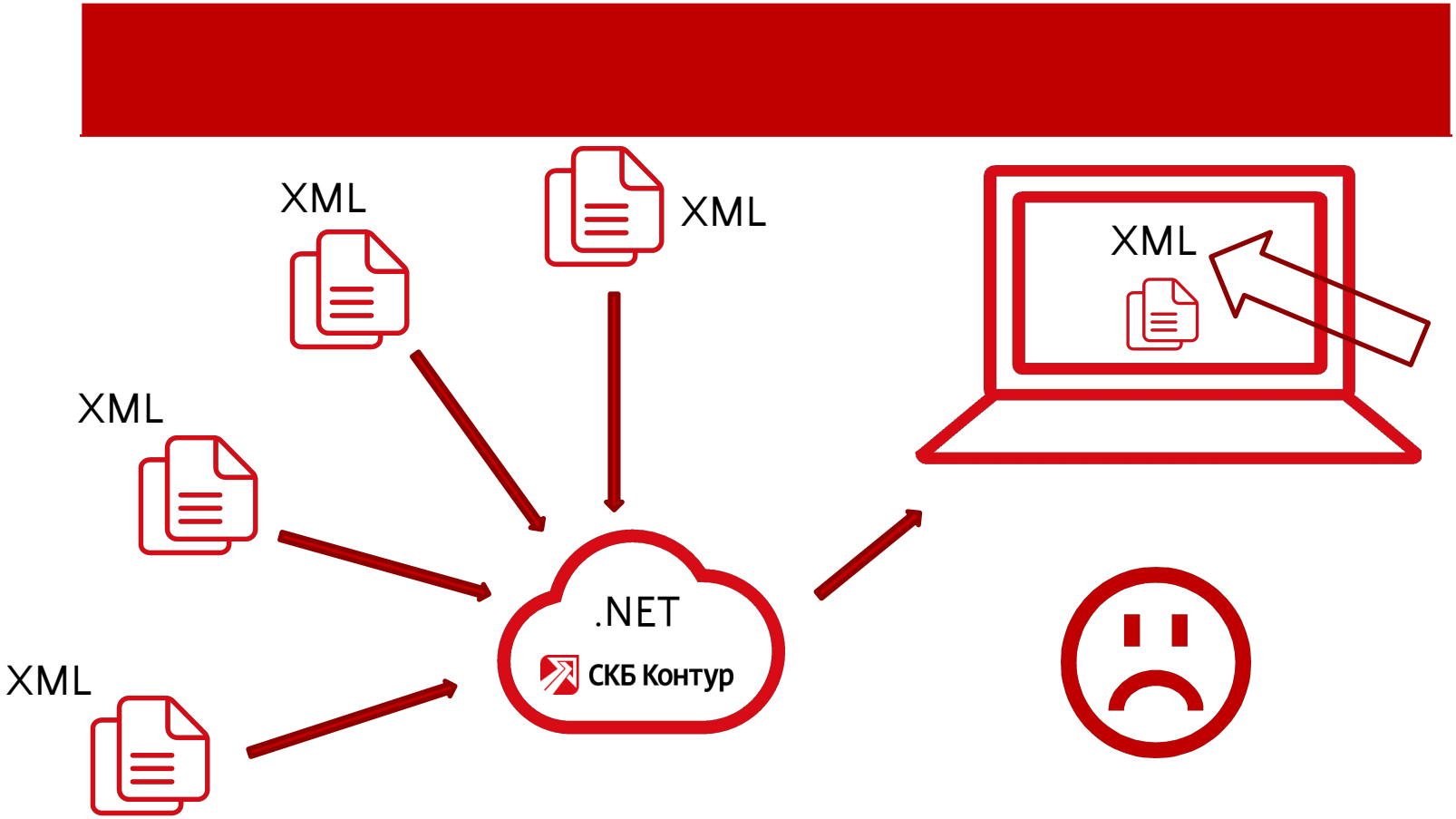
ИНТЕГРАЦИЯ ВИРТУАЛЬНЫХ МАШИН .NET И JAVA

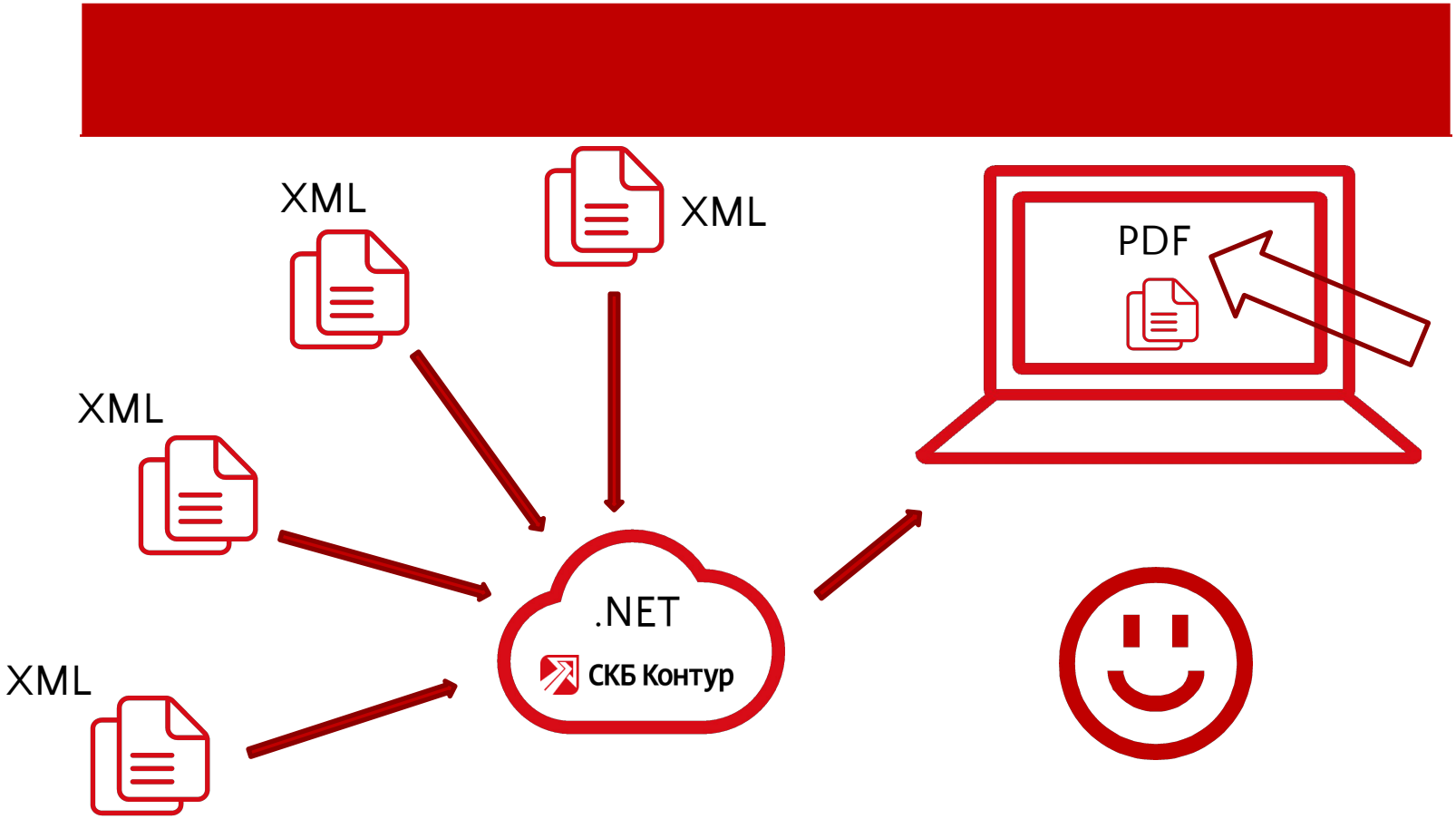
.NET MEETUP

Мария Телятникова
Григорий Кошелев

Екатеринбург, 28 февраля 2017

- Руковожу командой разработки (Java)
- > 5 лет пилю Java энтерпрайз
- (но слежу за развитием платформы .NET)


XML
XML
XML
XML
XML
.NET
СКБ Контур


XML
XML
XML
XML
PDF
.NET
СКБ Контур

Apache FOP

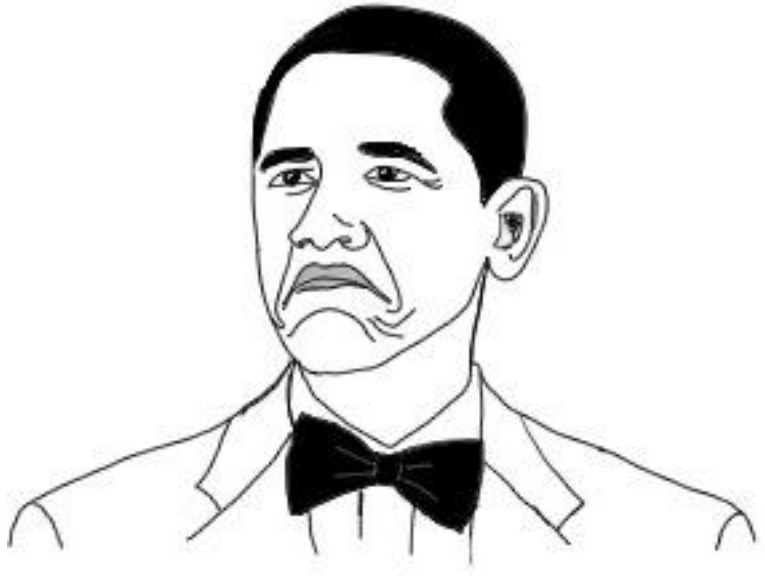
- Начало разработки – неизвестно
- Передана в ASF в 1999 году
- Apache FOP 1.0 – 21.07.2010
- Apache FOP 1.1 – 20.10.2012
- Apache FOP 2.0 – 03.06.2015
- Apache FOP 2.1 – 14.01.2016

- Переписать всё на C#
- Использовать кросс-компиляцию*
- Использовать из Java

(*) IKVM

?

- Переписать всё на C#
- Использовать кросс-компиляцию
- **Использовать из Java**

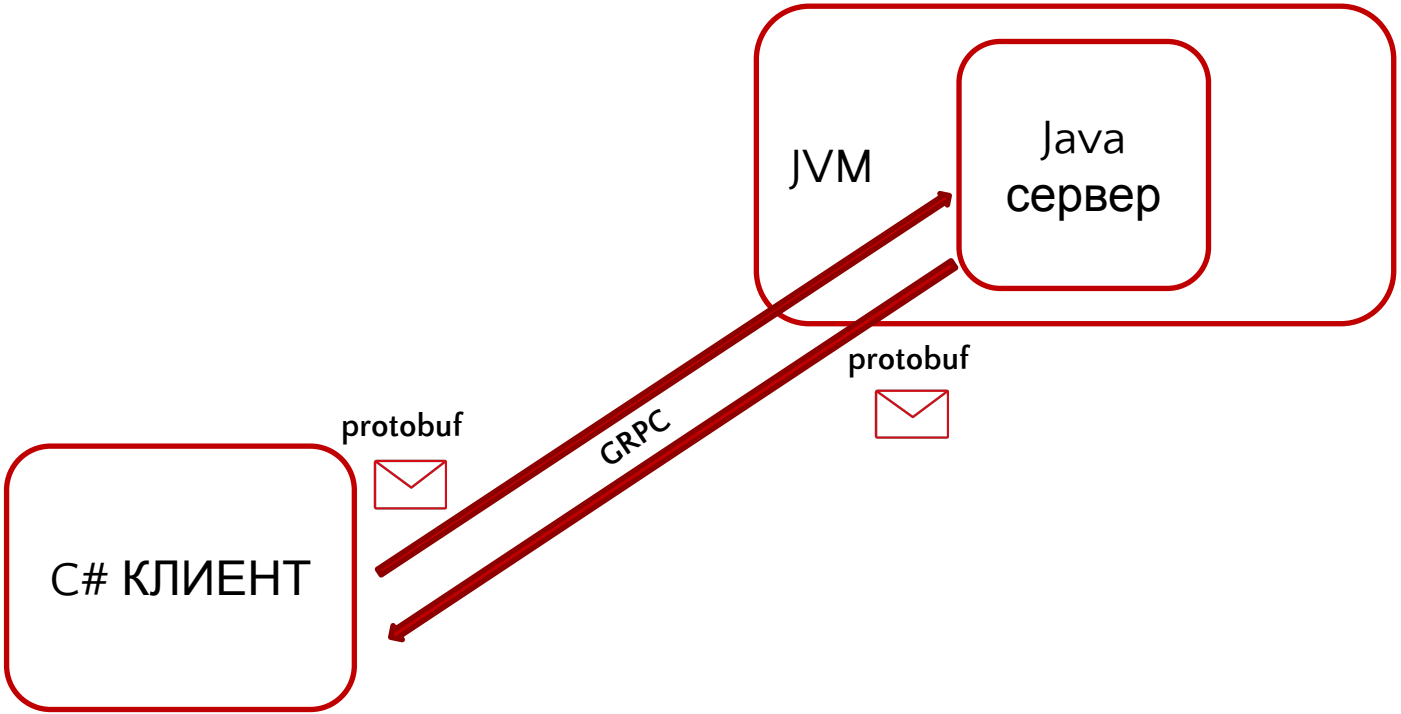


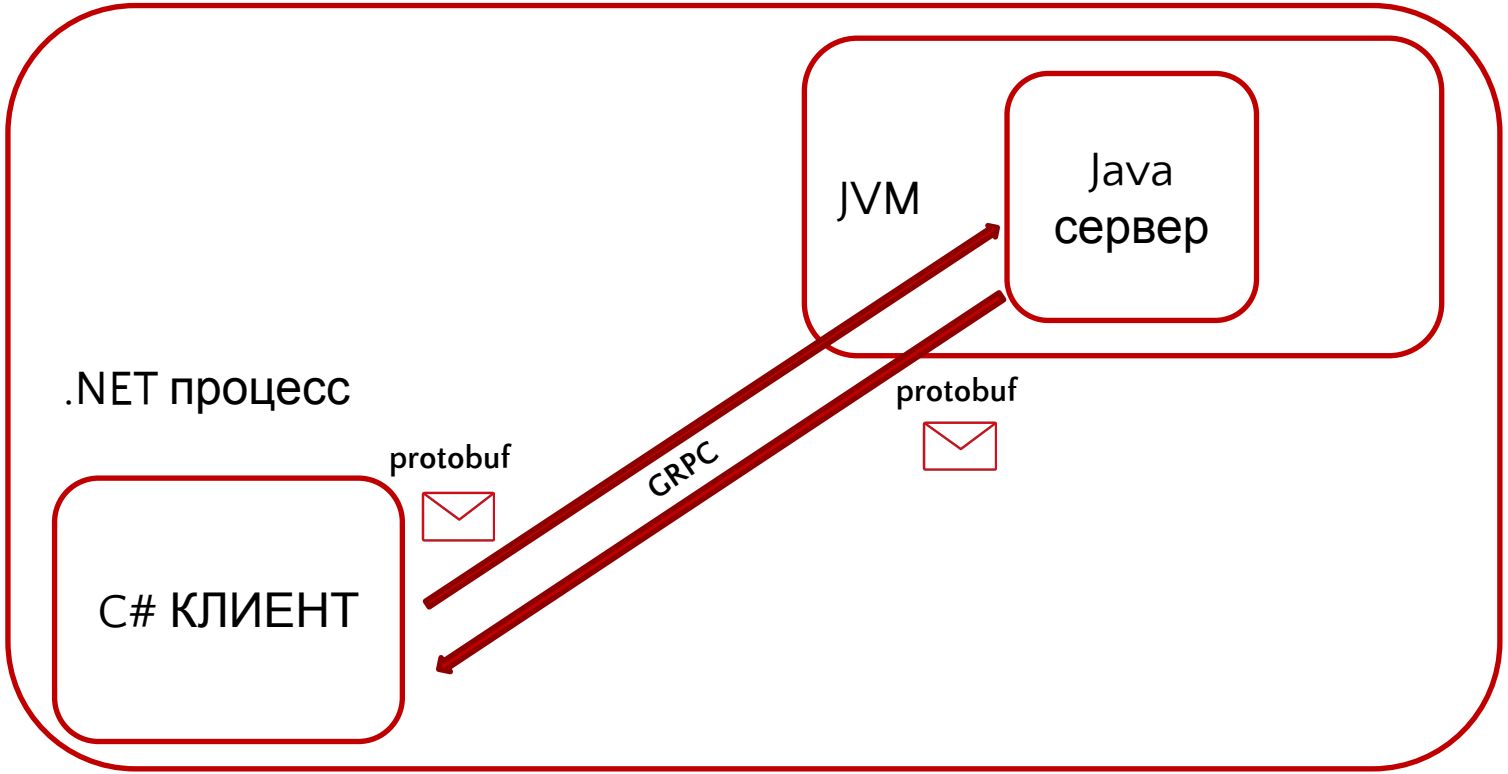
- HTTP REST
- Protobuf
- gRPC

- Библиотека для удалённого вызова процедур
- Поддерживаются: C/C++, Node.js, Python, Ruby, Objective-C, PHP, C# и Java

- Первый публичный релиз – 26.02.2015
- gRPC 1.0.0 – 19.08.2016
- gRPC 1.1.2 – 08.02.2017

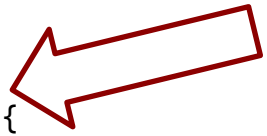
- Protobuf 3 – описание типов данных и сериализация
- HTTP/2 в качестве транспорта
- Кодогенерация плагином для Protobuf


JVM
Java
сервер
protobuf
protobuf
GRPC
C# КЛИЕНТ

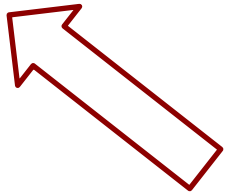

JVM
Java
сервер
.NET процесс
protobuf
protobuf
GRPC
C# КЛИЕНТ

```
JavaVM *jvm;
JNIEnv *env;
JavaVMInitArgs args;
JavaVMOption* options = new JavaVMOption[1];
options[0].optionString = params;
args.nOptions = 1;
args.options = options;
args.version = JNI_VERSION_1_6;
args.ignoreUnrecognized = 0;
int result = JNI_CreateJavaVM(&jvm, (void**)&env, &args);
```

```
jclass programClass = env->FindClass("ru/kontur/Program");
jmethodID doSmthMethod = env->GetStaticMethodID(programClass, "doSmth",
"(ILjava/lang/String;)I");
jint intParam = …;
jstring stringParam = …;
jint result = env->CallStaticIntMethod(programClass, doSmthMethod, intParam,
stringParam);
```



```
package ru.kontur;
public class Program {
    public static int doSmth(int x, String str) {
        return 0;
    }
}
```



```
JavaVmWrapper vm;
JniEnvWrapper env;
unsafe {
    IntPtr envP; IntPtr vmP;
    CreateJavaVm(&vmP, &envP, vmArgsP);
    vm = new JavaVmWrapper(vmP);
    env = new JniEnvWrapper(envP);
}
var classPtr = env.FindClass("ru/kontur/fop/service/FopServiceServer");
var constructor = env.GetMethodId(classPtr, "<init>", "(I)V");
var runMethod = env.GetMethodId(classPtr, "run", "()V");
var obj = env.NewObject(ClassPtr, constructor, port);
env.CallObjectMethod(obj, runMethod);
```

- Intel Core i7-4771, 3.5 ГГц
- 16 ГБ ОЗУ
- Windows 7 Professional x64
- Java 8 (1.8.0_121)
- .NET 4.5

- 10000 XML-документов (~ 1.53 ГБ)
- 1000 * 5 – на разогрев
- 9000 – на замеряемый прогон

-Xmx512m -Xms512m -Xss1m -XX:-TieredCompilation

.NET процесс

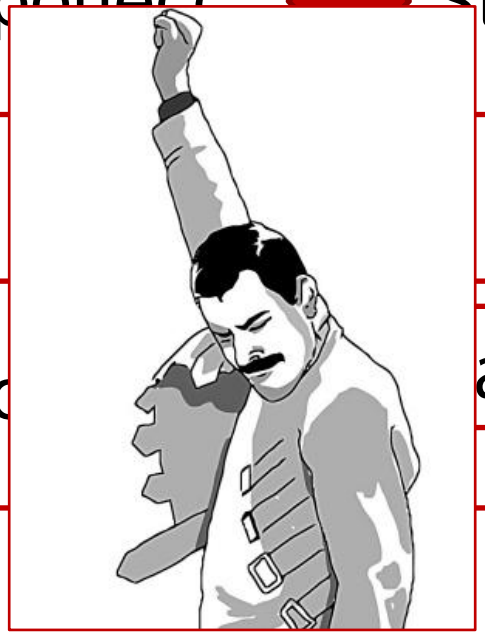
Standalone Java

59 468 ± 533 мс

.NET про

а

60 496 ± 716 мс



?

- Как запустить
- Что нужно знать

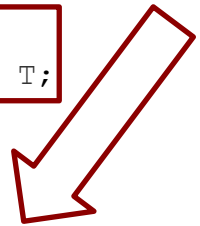
```
[DllImport("kernel32.dll", EntryPoint = "LoadLibraryW")]
private static extern IntPtr LoadLibrary([MarshalAs(UnmanagedType.LPWStr)] string name);

[DllImport("kernel32.dll", EntryPoint = "FreeLibrary")]
private static extern bool FreeLibrary(IntPtr hModule);

[DllImport("kernel32.dll", EntryPoint = "GetProcAddress")]
private static extern IntPtr GetProcAddress(IntPtr hModule,
        [MarshalAs(UnmanagedType.LPStr)] string name);
```

- System.Runtime.InteropServices
- Предоставляет методы для работы с неуправляемой памятью и кодом

```
protected T GetDelegate<T>(string procName) where T : class
{
    return Marshal.GetDelegateForFunctionPointer(
            GetProcAddress(hModule, procName), typeof(T)) as T;
}
```



```
public unsafe void CreateJavaVm(IntPtr* vmP, IntPtr* envP, IntPtr vmArgsP)
{
    var result = GetDelegate<CreateJavaVm>("JNI_CreateJavaVM")(vmP, envP, vmArgsP);
    /* */
}
```

```
unsafe delegate int CreateJavaVm(IntPtr* pvm, IntPtr* penv, IntPtr args);
```

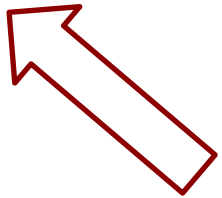
```
[StructLayout(LayoutKind.Sequential)]
public struct JniNativeInterface {
    /* ... */
    private readonly IntPtr GetVersion;
    private
    private readonly IntPtr FindClass;
    /* ... */
}
```

```
JniNativeInterface obj =
    (JniNativeInterface)Marshal.PtrToStructure(
        structPtr,
        typeof(JniNativeInterface));
```

```
struct JNINativeInterface {
    /* ... */
    jint (JNICALL *GetVersion)(JNIEnv *env);
                const jbyte *buf,
                jsize len);
    jclass (JNICALL *FindClass)
        (JNIEnv *env, const char *name);
    /* ... */
}
```

```
[StructLayout(LayoutKind.Explicit)]
public struct JValue
{
    [FieldOffset(0)] private bool booleanValue;
    [FieldOffset(0)] private byte byteValue;
    [FieldOffset(0)] private char charValue;
    [FieldOffset(0)] private short shortValue;
    [FieldOffset(0)] private int integerValue;
    [FieldOffset(0)] private long longValue;
    [FieldOffset(0)] private float floatValue;
    [FieldOffset(0)] private double doubleValue;
    [FieldOffset(0)] private IntPtr pointerVlaue;
}
```

```
typedef union jvalue {
    jboolean z;
    jbyte b;
    jchar c;
    jshort s;
    jint i;
    jlong j;
    jfloat f;
    jdouble d;
    jobject l;
} jvalue;
```



```
jclass (JNICALL *FindClass) (JNIEnv *env, const char *name);

delegate IntPtr FindClass (
    IntPtr env,
    [MarshalAs(UnmanagedType.LPTStr)] string str);

delegate IntPtr FindClass (
    IntPtr env,
    [MarshalAs(UnmanagedType.LPArray)] byte[] utf);
```

```
Encoding.UTF8.GetBytes(str)
;
```

U+0073 Латинская буква «s» (0x73) ✔

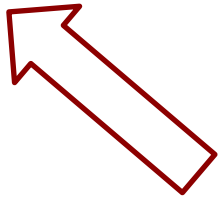
U+041A Кириллическая буква «К» (0xD0 0x9A) ✔

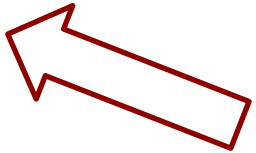
U+0BF5 Символ года на тамильском «๵» (0xE0 0xAF 0xB5) ✔

U+2?218 Китайский иероглиф «□» (0xF0 0xA6 0x88 0x98)

- Ручное управление объектами, размещёнными в нативной памяти
- Оборачиваем в IDisposable-обёртки

```
public void InjectClass(string className, byte[] classBytes)
{
    var loaderClass = new JavaClassLoaderClass(env);
    var loader = loaderClass.GetSystemClassLoader();
    var loadedClass = env.DefineClass(
                className,
                loader,
                classBytes,
                classBytes.Length);
    loadedClass.Dispose();
}
```





- Запустили виртуальную машину Java внутри .NET процесса
- Реализовали обмен данными между программами на Java и на C#
- Получили готовый работающий прототип решения задачи конвертации документов

- Загружать Java-библиотеки разных версий
- Попробовать передачу данных через нативную память напрямую
- Проанализировать поведение виртуальных машин .NET и Java в одном процессе
- Понять, куда это богатство ещё применить
- Сделать C#-обёртку удобнее

Посмотрели на Performance Counter сборки мусора по поколениям (разница между значением счётчика после прогона и перед прогоном)

Поколение	???	???
0	159	157
1	10	157
2	3	3

Почему так – будем думать и разбираться

Поколение	Inside	Standalone
0	159	157
1	10	157
2	3	3

ВОПРОСЫ?

Мария Телятникова
Григорий Кошелев

- Apache FOP https://xmlgraphics.apache.org/fop/
- IKVM https://www.ikvm.net/
- Protobuf https://developers.google.com/protocol-buffers/
- gRPC http://www.grpc.io/
- JNI http://docs.oracle.com/javase/8/docs/technotes/guides/jni/spec/jniTOC.html
- Interop Marshalling https://msdn.microsoft.com/ru-ru/library/eaw10et3(v=vs.110).aspx
- Marshal https://msdn.microsoft.com/ru-ru/library/system.runtime.interopservices.marshal(v=vs.110).aspx